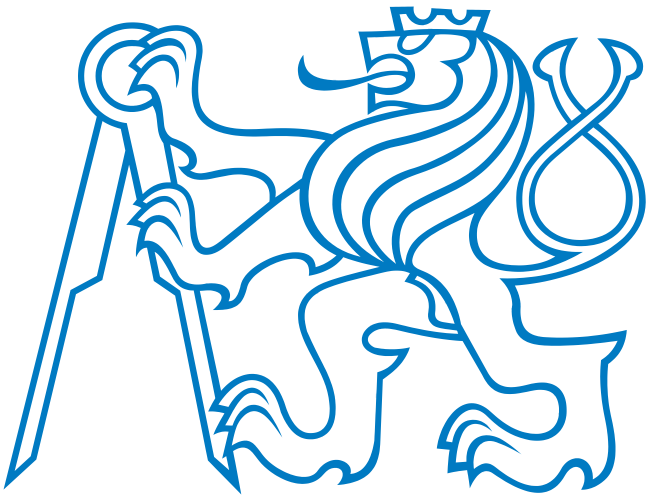
Czech Technical University in Prague

Faculty of Electrical Engineering

BACHELOR'S THESIS



Jiří Valtr

External Vision-Based Localization System for Mobile Robots

Department of Computer Science

Supervisor: **Ing. Petr Vaněk**

Prague, 2014

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

.....
Podpis autora práce

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Jiří V a l t r

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: External Vision-Based Localization System for Mobile Robots

Guidelines:

1. Study the external vision based localization system [1].
2. Implement the localization system for Android operating system.
3. Propose and design simple user interface for using the localization system with handheld devices such as smartphones and tablets.
4. Extend the localization system for using several cameras (devices) to provide a larger coverage of the robot operational environment.
5. Develop an easy-to-use procedure for calibation and deployment of the systém with several cameras.
6. Experimentaly verify performance of the localization system in terms of computational requirements and achievable precision for different hardware platforms.
7. Discuss possible improvement of the localization precision using multiple views of the localization pattern provided by several cameras.

Bibliography/Sources:

[1] Faigl, J. - Krajník, T. - Chudoba, J. - Přeučil, L. - Saska, M.: Low-Cost Embedded System for Relative Localization in Robotic Swarms. In Proceedings of 2013 IEEE International Conference on Robotics and Automation. Piscataway: IEEE, 2013, p. 985-990.

Bachelor Project Supervisor: Ing. Petr Vaněk

Valid until: the end of the winter semester of academic year 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, October 25, 2013

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jiří V a l t r

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Kamerový lokalizační systém pro mobilní roboty

Pokyny pro vypracování:

1. Seznamte se s kamerovým lokalizačním systémem pro mobilní roboty [1].
2. Implementujte lokalizační systém pro operační systém Android.
3. Navrhněte a vytvořte jednoduché uživatelské rozhraní pro použití lokalizačního systému na přenosných zařízeních jako jsou chytré telefony a tablety.
4. Rozšiřte lokalizační systém pro více kamer (zařízení) tak, aby bylo možné lokalizovat roboty v rozlehlejších prostředích.
5. Vytvořte snadno použitelný postup pro kalibraci a nasazení systému s více kamerami.
6. Experimentálně ověřte výkon lokalizačního systému s ohledem na výpočetní požadavky a dosažitelnou přesnost lokalizace s využitím různých hardwarových platformem.
7. Diskutujte možnosti zvýšení přesnosti lokalizaci s využítím více pohledů na lokalizační obrazec z více kamer.

Seznam odborné literatury:

[1] Faigl, J. - Krajník, T. - Chudoba, J. - Přeučil, L. - Saska, M.: Low-Cost Embedded System for Relative Localization in Robotic Swarms. In Proceedings of 2013 IEEE International Conference on Robotics and Automation. Piscataway: IEEE, 2013, p. 985-990.

Vedoucí bakalářské práce: Ing. Petr Vaněk

Platnost zadání: do konce zimního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 10. 2013

Abstract

This thesis deals with the robot localization problem. It presents a system based on the camera localization system Whycon, which is adapted for use with handheld devices running Android operating system. The system is extended to take advantage of several such interconnected devices, each with its own dedicated camera, providing the localization system with multiple additional points of view. Deployment of the system is easily done by installing an application on each of the devices. The application has a simple user interface, which allows the user to carry out all the preparation procedures necessary for the system to be able to incorporate the device in the localization. The multiple points of view, resulting from the usage of several devices, give the system the potential for increased spatial coverage and precision. This thesis describes the challenges of dealing with several points of view in a camera-based localization system, both while deploying it and during its operation. It proposes convenient routines, which can be used to increase performance and efficiency of the system.

Abstrakt

Tato bakalářská práce se zabývá problémem lokalizace robotů. Představuje systém založený na kamerovém lokalizačním systému Whycon, který je přizpůsoben pro použití na přenosných zařízeních s operačním systémem Android. Systém je rozšířen tak, aby využíval několik takovýchto vzájemně propojených zařízení, každé s vlastní kamerou, která systému poskytují několik dalších úhlů pohledu. Pro použití systému je třeba pouze na každé zařízení nainstalovat příslušnou aplikaci. Ta obsahuje jednoduché uživatelské rozhraní, pomocí kterého je možné provést všechny úkony, potřebné k přípravě zařízení k využití v rámci lokalizačního systému. Několik pohledů, dostupných díky použití několika zařízení, dává systému potenciál k pokrytí většího prostoru a ke zvýšené přesnosti. Tato práce popisuje problémy vyvstávající při práci s kamerovým lokalizačním systémem s více pohledy, jak při jeho instalaci, tak během provozu. Představuje praktické postupy, pomocí nichž lze dosáhnout vyššího výkonu i lepší efektivity celého systému.

Acknowledgements

I wish to thank my supervisor Ing. Petr Vaněk for his inspiring suggestions and patient guidance. I further wish to thank my family for their never ending support throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Localization in mobile robotics is one of the key problems to be solved and is oftentimes a necessity for the robot to fulfil its purpose. If we compare mobile robots and their need of localization with humans, we find many analogies in the kinds of problems both robots and humans have to deal with. Some of the tasks, which such an individual may have to perform—like transportation or locating of another object—are so closely tied with the localization problem, they would not make any sense without it. Before one travels to a certain destination, one needs to know, where his starting point is, so he can choose an applicable route. Similarly, to be able to tell another object's exact location, an individual has to be aware of its own position with comparable precision. To solve this problem, humans are usually able to employ several different sources of information, so they can confirm the results, as well as increase the precision or robustness of the localization. Robots on the other hand, mostly have a limited number of mechanisms to do so.

There are many ways to determine a position, generally distinguished by the used sensors and overall approach. One of such approaches is to take advantage of an external device, which overlooks the general area, where the subject of localization is expected to be placed. Such a device then provides the subject with its position with respect to a certain landmark or even with respect to a defined origin of a coordinate system. This thesis deals with such an arrangement, in which there are several devices overlooking an area, where the robot is expected to appear. These devices are equipped with cameras, which allow them to evaluate the location of the robot within their view. Then they use wireless communication to send the results to the robot, so it can use them for its own purposes.

It is a goal of this thesis to propose such a system, that would be easily deployable on hardware already in place, so that it can be tested and used by developers

without much hassle. This involves creating a simple user interface for using the localization system with handheld devices. Therefore, the proposed system is designed to work on devices running the Android operating system, which are cheap, commonly available and have the necessary hardware, such as the camera and the wireless communication facilities. It is an extension of the system proposed by Tomáš Krajník, et al. [1] and by Jan Faigl, et al. [2].

1.1 Organization of the Thesis

The thesis is organized in the following way. Below this introduction, Chapter 2 introduces the reader to the problems of localization and provides a simple overview of the commonly used solutions. Chapter 3 provides a brief introduction to computer vision and camera calibration. Chapter 4 describes the basic method of localization employed by the system described in this thesis. Chapter 5 deals with the usage of several cameras, which extend the capabilities of the system. And finally, Chapter 6 presents some experimental results and achievements of the proposed system.

Chapter 2

Localization Methods

The localization problem is one of the key problems in any advanced robotic task, especially when considering mobile robots. By localization I mean the process of determining an exact position of all relevant parts of the robot in relation to its surroundings. Since it is such an important and frequently encountered problem, there are many known ways of dealing with it. In this chapter, I present an overview of available localization techniques and their basic categories, to put the chosen approach into perspective. I also show common use cases for each category to illustrate a typical scenario, in which it makes sense to use a solution similar to the one described in this thesis.

2.1 Taxonomy of Localization Methods

Localization methods can differ in lots of aspects, be it precision, its variability in time, area coverage, purpose, or even price of deployment. They can be divided to several different groups, using these aspects as criteria.

2.1.1 Relative and Absolute Localization

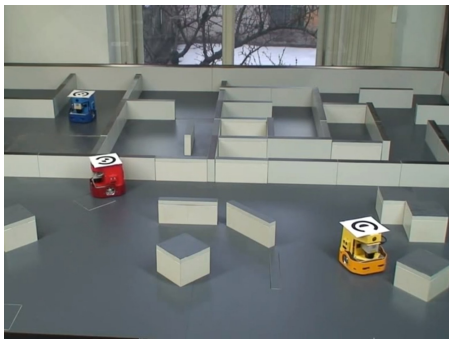
Based on the character of localization process, it is distinguished between *relative* and *absolute localization* [3].

In *relative localization*, the change of position against a previous step or frame is measured. These relative changes in position are then stacked one after another, reconstructing the previous movements of the robot and therefore determining the

current position of the robot. To describe this position in relation to outside world, the *a priori* knowledge of the starting point location is required. Precision of such systems declines in time with the growing number of measurements each relying on correctness of the preceding one. Every next steps brings a new, however small, inaccuracy into the system irreversibly and furthermore worsens the overall precision. Because of this, it only makes sense to use such localization methods over short periods of time, or more precisely, over a small number of measurements.

In *absolute localization* it is not a concern, as each measurement is completely independent on the previous one. The precision of this localization is therefore constant, although it is not necessarily better than the precision of relative system. It is also usually more difficult to set up both technically and financially and it is often limited to a certain area. For these reasons, it is more often used for indoor applications. Figures 2.1a and 2.1b show examples of such applications.

(a) RoboCup robotic football competition



(b) SyRoTek (System for Robotic E-learning)

Figure 2.1: Absolute localization applications

To increase precision and robustness of the localization a combination of both approaches described above is often used in practical applications. A typical scenario could be a mobile robot, which uses a lower frequency measurements from an absolute localization system to eliminate the cumulative error and a higher frequency measurement from a relative localization system to increase the operation speed, as it does not have to wait for the next absolute measurement to take place and can make do with the relative one until such time. In this case, the localization system described in this thesis would be well suited to play the part of the lower-frequency reference, because it uses the absolute localization approach. Also, its suggested deployment on real world of the shelf devices implies it will mostly be limited to frame rates under 30 FPS, so its usage for high frequency measurements can prove to be problematic.

Another situation, in which the absolute localization system could be used coupled with a relative one, might be in development, when the absolute system provides a reference frame. Such a reference frame may then be used to measure precision, while testing newly developed localization system.

2.1.2 Global Localization and Tracking

Based on the knowledge of the starting position, a different problem is solved by the localization system [4].

During *tracking*, an assumption can be made that the initial position of the robot is known. This knowledge is used during the localization to make estimates of possible robot position in the next step. However, if the system relies on these estimates too greatly, it can lead to an unrecoverable fault in localization.

Global localization does not need any previous knowledge. It searches the whole area of coverage, so it can even be used to solve the robot wake-up problem or the kidnapped robot problem.

The system described here is capable of localizing the target, without any *a priori* information. It does, however, use the knowledge of the position from the previous measurement to determine the starting point of search in the current one, increasing performance greatly.

2.1.3 Active and Passive Localization

Based on the ability of the localization system to influence the control of the robot, it can be distinguished between *active* and *passive localization* [3].

In *passive localization*, the location estimate is based only on the feed of data from a sensor. *Active localization*, on the other hand, allows the localization system to directly influence or even completely take over the control of the robot when necessary. This may happen, for example if a sensor is installed on a moving part of the robot, such as an arm.

Passive localization is the more common of the two, active localization being used only in special cases.

2.2 Probabilistic Localization

When facing the localization problem, there is often more than one source of information. In most cases, there are several sensors or systems, providing use with different kinds of information of inconsistent precision and varying regularity. There is therefore a need to merge this data to be able to use them for robot localization. The most common approach to the data fusion is the probabilistic localization [3, 5].

According to the probabilistic approach, the sensor findings are random variables representing location estimates. Also, the probability density is defined as a function assigning every possible location in the working space a probability, that it is the real robot's position. Based on the above, the robot localization problem then becomes a problem of finding such an estimate of probability density, which best fits the probability density of the robot's actual position. The sought probability density is called *belief* [5].

The goal of the probabilistic localization is to keep the belief as close as possible to the probability density of the robot's actual location. To achieve this, the belief is updated after each received set of the results from the sensors. The first type of update, made based on the resources of relative localization, is called *prior belief* or *prediction*. In this step, the location is actually predicted based on the previous robot location and the most recent sensor data. The second type of update is called *posterior belief* or *correction*, as it corrects the location estimate based on the data from absolute sensors.

The *initial belief* is the inceptive value of the belief, even before there are any data from the sensors. It can be the case, that the original position of the robot is known, in which case the belief function has a single peak. On the other hand, if the starting position of the robot is not known, the belief has uniform distribution over all the locations. There might even be cases, when the initial belief is none of the above, such as if the robot has several possible starting locations, but the first two cases are the most common ones.

To be able to correctly calculate the belief, there needs to be a model, which describes, how to use the sensor data z_i, acquired in step i, to correct the estimate of location l_i. This *perceptual model* expresses the conditional probability $P(z_i|l_i)$ of measuring the last incoming data, given the robot is at the previous location (i.e. the one being corrected). The perceptual model can be represented either by an expression, describing how to calculate the value of $P(z_i|l_i)$, or it can be pre-calculated for each possible combination of z and l [4].

There are several ways to represent the perceptual model. The discrete ways

include a grid of probabilities, topological graphs and Monte Carlo localization, while the most common continuous way lies in Kalman filters. Description of advantages and differences between the above mentioned methods can be found in [3, 4, 5].

2.3 Common Localization Techniques

To conclude this chapter, I hereby present some of the common examples of localization techniques, divided into the relative ones and the absolute ones.

2.3.1 Relative Localization Techniques

Dead Reckoning is the simplest method for localization estimates, based solely on the speed, direction of movement and time passed since the last known position. This old method was originally used by sailors and later by pioneering air plane pilots. As a common relative localization method, this technique accumulates additive error and therefore is not usable in the long term on its own. However, it is a good complimentary method when used with a precise absolute localization technique, such as when the ancient sailors used it coupled with celestial navigation [3].

Odometry is a method based on estimating the change in wheeled robot's position by counting the number of revolution of each wheel. The counting is done by encoders, which translate rotation movement into electric signal. There are lots of types of encoders in use, the more widely used being the optical encoders, the brush encoders, potentiometers and the magnetic encoders. The simple optical encoders are considered to be generally the best [6].

Same as dead reckoning, odometry is subject to cumulative errors. The basic additive error, caused by chaining too many measurements, is even worsened by inaccuracies stemming from the basic premise of odometry, that the revolving motion of the wheels precisely translates to the linear motion of the robot. Lots of factors may prevent this from being true. Typically the disparity is caused by unequal size and shape of the wheels, finite resolution and sampling frequency of the encoder, or random effects of the environment, such as uneven ground or loss of adhesion.

Inertial Navigation uses gyroscopes and accelerometers to measure the rate of rotation and acceleration of the robot. By integrating these values, it is possible to calculate speed and position estimates. However, because the measurements are

calculated by integration, the results drift over time, and thus the errors increase without bound [4]. A popular apparatus, which provides a simple way for robots to utilize inertial navigation, is the so-called Inertial Measurement Unit, consisting of three axis angular turning rate detector and three axis accelerometer [7]. Lately, it finds particularly extensive use in quadcopters.

2.3.2 Absolute Localization Techniques

Landmark Based Localization relies on detection of landmarks. Landmarks are any features in the environment that a robot can detect. When landmarks are detected, they are matched with *a priori* information known about the environment to determine the position of the robot. (Even when there is no *a priori* information about the environment, it is possible to localize the robot using the Simultaneous Localization and Mapping (SLAM) techniques [8].)

Landmarks can be divided to active and passive. *Active landmarks* or *beacons* actively broadcast location information. There are two methods commonly used to determine the absolute location of the robot: triangulation and trilateration. Triangulation uses distance and angles to several active landmarks, while the trilateration only uses distances. The most well known example of such a system is the Global Positioning System (GPS), which uses trilateration techniques to determine latitude, longitude and altitude [4].

Active landmark localization is usually very precise and not very computationally intensive. Disadvantages of using active landmarks are the cost, potential interference or unavailability of the signal in some parts of the environment, and the sole need of preparing the environment for the robot.

If the landmarks are not actively transmitting signals, they are called *passive landmarks*. Therefore, it is the robot, which must actively search for the landmarks within its environment. The detection of the landmarks differs according to the type of sensor used, the most common way of detection being visual—using cameras.

Passive landmarks may further be divided to natural and artificial. *Natural landmarks* are detectable shapes, which are natural parts of the environment. Indoors, these could be lights, corners of the room, windows or doors. In the exteriors, typical landmarks are trees, roads or road signs. To find these landmarks in the image, feature descriptor extraction techniques are employed, such as the Scale-invariant feature transform (SIFT) [9].

The lack of the necessity to prepare the environment is the greatest advantage

of natural landmarks and is the main reason the passive landmarks are widely used in outdoors robot navigation. However, they are usually not unique, hard to detect and they may change unexpectedly.

Thus, it is often better to use *artificial landmarks*, such as bar codes and contrasting or coloured geometric figures. This method is used mostly indoors, where it is relatively easy to position the landmarks. To visually recognize it, sufficient lighting and direct view of the landmark is necessary, which can also be mostly expected indoors. Disadvantage of this approach lies in the limited ability to detect the landmark from a grater distance.

The localization system discussed in this thesis is precisely of this kind—it uses passive artificial landmarks in the form of contrasting geometrical figure coupled with visual sensing to perform absolute localization of mobile robots.

Chapter 3

Computer Vision

Computer vision is the enterprise of automating and integrating a wide range of processes and representations used for vision perception [10]. In general, computer vision aims at using cameras for analysing and understanding scenes in the real world [11]. It includes techniques for geometric modelling, cognitive processing, statistical pattern classification and image processing. Many of its approaches are mimicking practices commonly used by vision system of humans or animals.

A complete computer vision system should be able to deal with two kinds of problems [10]. Firstly, there are the problems solved by low-level processes, such as the ability to determine and quantify lightness, colour, range, etc. These are not always strictly technical abilities. For example, human colour perception allows us to recognize black object as black even in a complex scene, where it reflects more light than objects of other colours. Other examples of these low-level capabilities are specialized object perception capabilities, such as the human face recognition, or the ability to visually remember images based on relatively primitive descriptions. Computer vision must try and reinvent these basic talents of biological visual systems.

Secondly, a computer vision system should be able to use solutions of the low-level tasks to fulfil goals, make plans for future actions, create complex models based on the perceived objects and administer cognitive processes. These abilities are called high-level capabilities. They give a purpose to the actions performed by the vision system. A good example of such purpose is the main topic of this thesis—localization.

Before moving on to more immediate applications of computer vision to challenges dealt with in this thesis, a very brief introduction of the basic concepts in computer vision is presented below.

3.1 Important Image Characteristics

The most basic data, with which a computer vision system commonly works, is an *image* of a scene. An *image* is defined by integrating and sampling continuous real-world data in a spatial domain. It consists of an array of pixels, each combining a position $(x, y) \in \mathbb{Z}^2$ and a value u, the sample at position (x, y). The pixels at positions $(x, y) \in \mathbb{Z}^2$ form a regular rectangular grid [11].

Values u of the pixels may be single discrete integers in a predefined range, representing the intensity of light at that position, or they may be sets of several such values, which can for example represent the colour channels in an RGB image. These values are commonly stored as two or three dimensional arrays of numbers, where the location of the pixel corresponds with the position of the value in the array. Analysis of the values stored in such arrays is the basis of most computer vision tasks.

3.1.1 Image Structure

An important structure defining feature, which can be found in a majority of images, are *edges* or discontinuities in the sampled values of neighbouring pixels. Edges are important information for understanding the image and are also useful for simplifying the data representing the image. A convenient way to perform the simplification and map the image into an edge image is by interpreting the image intensity values as defining a surface having different elevations at pixel locations. In that interpretation, the first- or higher-order derivative can be used to map most of the intensities to values close to zero, only leaving pixels near the discontinuities (i.e. the edges) with intensities of great absolute values. It appears to be meaningful to detect edges at locations where the second order derivative define a zero-crossing [11].

An alternative to using discontinuities in spatial domain as a key to understanding the image structure lies in transforming the image from spatial to frequency domain. Once transformed, low frequencies represent homogeneous additive contributions to the input image, while high frequencies show local discontinuities such as edges or intensity outliers. Spectrum and phase of the image mapped to polar coordinates provide a convenient way to describe the image structure regardless of the absolute values of the pixel intensities.

3.1.2 Colour Representation

Colour is a perceptual phenomenon related to human response to different wavelengths in the visible electromagnetic spectrum (approximately 400 – 700 nm). In human vision, colour is one of the most important characteristics of the perceived scene. Further, it is useful or even necessary for powerful visual processing of the real world. Computer vision is therefore faced with the problem of describing and interpreting the colours in a scene.

Colour spaces are a way of organizing the colours perceived by a vision system. It happens that weighted combinations of stimuli at three principal wavelengths are sufficient to define almost all the colours humans can perceive. These wavelengths form a natural basis from which the colour measurement process can be described [10].

There are several bases commonly used in computer vision to represent colour. The most common is the RGB space, which is an additive colour model and which uses a multi-channel image with values $u = (R, G, B)$, where the R, G, B are intensities of the red, green and blue colour components respectively [11]. Other colour spaces include the HSI model (HSI = hue, saturation, intensity), the subtractive CMYK model or the CIE XYZ colour space. Conversions between colour spaces are possible and frequent, as different colour spaces are better suited for different use-cases.

Furthermore, while the colour is useful and very important in a lot of procedures, in tasks where it is irrelevant, it quickly becomes an unnecessary burden in terms of data volumes. In those situations, it makes sense to just use grayscale images, which not only reduces the amount of data by two thirds, but it even simplifies a lot of the calculations performed above the data.

3.2 OpenCV

OpenCV is an open source computer vision library written in C and C++ with interfaces for many other programming languages. It is designed to provide a simple to use computer vision infrastructure, which helps in building sophisticated vision applications. It also contains general purpose machine learning functions, which are highly useful for many vision tasks [12].

The localization system, which is a main topic of this thesis, uses OpenCV in several ways. Firstly, it uses it as an interface for the camera, which is connected to the device running the system. It takes advantage of the easy to use methods for

obtaining frames from the camera in a computer friendly form of two dimensional arrays (matrices).

Secondly, OpenCV is used as a convenient mathematics library. It offers various specialized data structures simplifying vector and matrix calculations. There are handy and effective functions for a lot of standard operations, such as functions concerning eigenvalues and eigenvectors.

And lastly and most importantly, OpenCV's algorithms are used to create an ideal pinhole camera model for the camera the system is using. Each frame captured by the camera is undistorted according to its intrinsic parameters, allowing for more precise localization. Then a transformation is done through the pinhole model, which assigns the projected point on the image plane a location in a real world three dimensional space. All of that is done with the help of OpenCV. Besides, convenient methods for camera calibration provided by the library are used to obtain the previously mentioned intrinsic parameters.

3.3 Camera Calibration

The information a camera has about the scene it captures is the intensity of pixels arranged in a two dimensional array. When an object is recognized in the image, only its two dimensional location in the image plane is known. For the purposes of the localization system, an exact position of an object in a three dimensional space is needed. It is therefore necessary to somehow associate a position in the two dimensional grid, which the camera sees, with an actual real world three dimensional position. This can be achieved by camera calibration and 3D reconstruction. By camera calibration I mean obtaining the four intrinsic parameters, which encompass focal length, image format and principal point position, as well as radial and tangential distortion coefficients.

The chosen approach uses a so-called pinhole camera model, where a scene view is formed by projecting points from the real world three dimensional space into the image two dimensional grid of pixels using a perspective transformation [13]. This transformation can be described with the following equation:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \, [\mathbf{R}|\mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1}$$

where s is the distance of the projected point from the focal plane of the camera, (u, v) are the coordinates of the projected point in pixels, $[\mathbf{R}|\mathbf{t}]$ is a matrix of

extrinsic parameters, which is a joint rotation-translation matrix describing the position of the object in front of a still camera or vice versa, (X, Y, Z) are the coordinates of the projected point in the world coordinate space and $\mathbf{A}$ is a matrix of intrinsic parameters, also known as camera matrix, which has the following shape:

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where (c_x, c_y) are the coordinates of the principal point, which is usually at the centre of the image and f_x, f_y are the focal lengths expressed in terms of pixels. If (x, y, z) are considered to be the coordinates of the projected point already transformed to a coordinate system fixed with respect to the camera, then the transformation above is equivalent to the following (if $z \neq 0$):

$$u = f_x \frac{x}{z} + c_x \tag{3.2}$$

$$v = f_y \frac{y}{z} + c_y \tag{3.3}$$

However, since real camera lenses always have some distortion, both radial and tangential parts of the distortion need to considered. Therefore the model has to be extended as follows:

$$x' = \frac{x}{z}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_1 \frac{xy}{z} + p_2 \left(r^2 + 2\left(\frac{x}{z}\right)^2\right) \tag{3.4}$$

$$y' = \frac{y}{z}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1 \left(r^2 + 2\left(\frac{y}{z}\right)^2\right) + 2p_2 \frac{xy}{z} \tag{3.5}$$

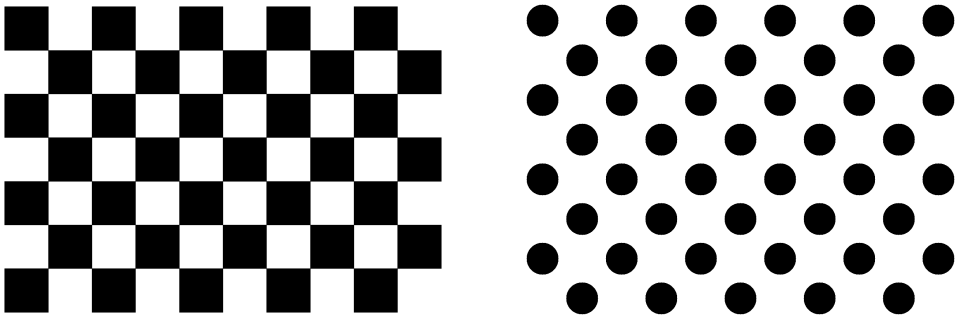
$$u = f_x x' + c_x \tag{3.6}$$

$$v = f_y y' + c_y \tag{3.7}$$

where k_1, k_2 and k_3 are radial distortion coefficients, p_1 and p_2 are tangential distortion coefficients and $r^2 = \left(\frac{x}{z}\right)^2 + \left(\frac{y}{z}\right)^2$.

Since my application has to deal with a lot of types of cameras, all with slightly different optics, I needed a unified way to determine these parameters. OpenCV library provides convenient methods for camera calibration [12, Chapter 11] using a black and white chessboard pattern or either a symmetric or asymmetric circles pattern, such as the ones seen in Figure 3.1. This allowed a creation of a simple interface for users to be able to calibrate their cameras in three easy steps.

1. Printing (or otherwise obtaining) the provided pattern.



(a) Chessboard pattern

(b) Asymmetric circles pattern

Figure 3.1: Calibration patterns

2. Capturing several pictures of the pattern from different perspectives with their camera. (Approximately ten shots is usually enough.)
3. Running a script that takes in the captured images of the pattern and calculates the desired parameters.

The calculated parameters are then automatically saved in the devices internal storage to be used in the main localization and tracking tasks.

My implementation of the calibration procedure works with the asymmetric circles calibration pattern, which, according to the OpenCV Documentation [14], requires data from less snapshots to form a well-posed equation system for the calibration.

Chapter 4

RobotTracker

The RobotTracker application for the Android operating system is based on the external localization system Whycon—a vision-based software for robot localization, described in [1], which has been created with the use of the OpenCV (Open Source Computer Vision) library. The core of RobotTracker application is essentially a Java port of the original algorithms written in C++, which is the native language of OpenCV, wrapped in user-friendly Android interface. It uses the same technique for detection of simple black and white circular patterns, which is robust to variable lighting conditions and achieves sub-pixel precision. Other key advantages of the chosen approach are its resolution independence and extensibility to multiple target detection [1]. However, it is not simply a mirror of the original code in another programming language. RobotTracker is also an extension of the project, trying to provide options for usage of several cameras to extend the spatial coverage of the system. Furthermore, it is aiming to take full advantage of the Android platform ecosystem, which gives a strong base of in place devices, so that many users will not be forced to buy an expensive new hardware and also provides an established distribution system—the Google Play.

Some of the features of this combination of software and hardware are not easily determined. First of all, the quality of the cameras fitted to common Android devices such as phones or tablets is very varying and while there are ways to quantize the quality of these cameras [15], easily comparable results aren't commonly available. Secondly, the processing power differs greatly across the broad spectrum of devices, so the performance on each particular device is hard to determine without real world testing. Last, but not least, since the version of OpenCV for Android is just a Java Native Interface wrapper of the native OpenCV code, there are some extra tasks needed to be performed at each point, where the native libraries are used. That means extra work for the processor, which somewhat hinders the en-

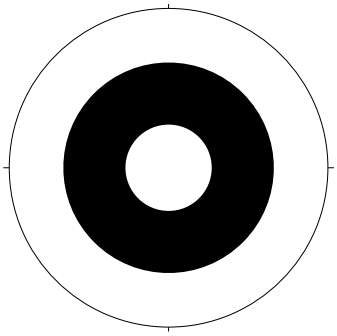


Figure 4.1: Circular pattern

tire system, as some of the performance is traded for the convenience of using the platform's common application programming language—Java.

The localization system described here works with a pinhole camera model. In reality, this means it is expected from this point further, that the camera has been calibrated and its intrinsic parameters and distortion coefficients are known. A possible way to obtain these parameters is described in the previous chapter, but it is not the only way. The application therefore allows for side loading of these parameters obtained by some other means, so that the user is not forced to repeat the process of calibration on each device. Whatever their origin, the calibration parameters are fundamental for the process of localization, which cannot be done without them.

4.1 Target Detection

The most basic task in the localization procedure is to locate the circular pattern in the image, in other words, determine the position of its centre in the image plane coordinates and its boundaries represented by maxima and minima in the x and y axes. The pattern consists of two concentric rings—the outer one white, the inner one black—and a white central disc (Figure 4.1). The detection itself uses a flood fill technique with on-demand thresholding [1].

In the first run, there is no *a priori* information, about the pattern's possible position in the image, so the picture needs to be searched starting from the top left pixel, until the pattern is found. However, while processing each succeeding frame, the last position of all the targets is known (except when the targets have not been found), so the system can start looking for them in these positions' surrounding areas and therefore likely decreasing the number of pixels that need to be checked.

Knowledge of the previous image also allows us to correctly set the threshold level, used to classify a pixel as either black or white, right away, whereas in the first run, in case of an unsuccessful detection, another try needs to be made with a different threshold. A situation, when the targets are not found at all, can be considered a worst case scenario, which leads to examination of every pixel of the image and execution of the flood fill method and the circularity checks on each black pixel segment present in the image.

The threshold is used as a reference, when determining, whether a pixel is black or white. The determination is done by comparing the brightness of the pixel with the threshold. In case the brightness of the pixel is lower than the threshold (i.e. the pixel is darker), the pixel is considered black. If it is the other way around, the pixel is considered white.

As a first step, the system is trying to find the outer black ring, so when a white pixel is detected, it can proceed to the next one. In case a black pixel is detected, a queue-based flood-fill method is initiated, which searches all the neighbouring pixels, until a border of white pixels is reached. Afterwards, various checks are performed on the resulting constrained segment of black pixels for a possible match with the outer black ring, most importantly minimum size and roundness check. No redundant pixel classification is done in further search should any of the checks fail, as the pixels are already marked as belonging to the current segment. If all these checks are passed, the algorithm starts the flood-fill method again, starting at the calculated centre of the black ring. This time it searches for white pixels inside a black border, expecting the inner white disc in the middle of the circular pattern to be found.

In case the white disc is successfully found as well, further testing takes place in order to recognize the examined part as the searched circular pattern and therefore a valid target. These tests include concentricity check and the two segments area ratio check. The frame is also made modified by painting all the pixels of the white inner disc of the current pattern black, which prevents multiple detection of the same target, in case more than one target is being localized.

Result of the procedures described above is a set of features, which are obtained during the process of detection. These include dimensions of the pattern in pixels, average brightness and, most importantly, position of the centre of the pattern in the two dimensional image plane coordinates. Obtaining the image plane coordinates of the target is the first key step in the process of solving the localization problem. Based on this entry data, further calculations can be made to determine the position of the target in real world.

4.2 Target Localization

Firstly, the position can be transformed from image plane coordinates to camera centric real world coordinates. This is done using the camera calibration parameters and the known pattern dimensions. The procedure—described in [1] and especially in [16]—involves calculating properties of the ellipse, as which the camera sees the circle, constructing the ellipse characteristic equation and transformation to canonical camera coordinate system.

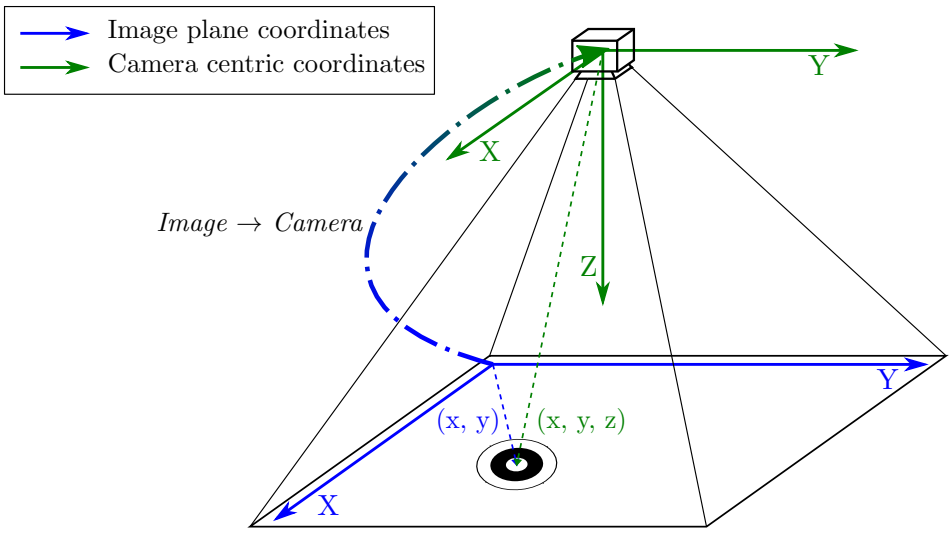


Figure 4.2: Transformation from image plane coordinates to camera centric coordinates

Output of the transformation are coordinates in a three dimensional coordinate system with origin at the principal point of the camera. A result like this may be sufficient in certain scenarios. Supposing the exact position of the camera (i.e. of its principal point) is known, the coordinate system belonging to it can be used as a global reference. This approach is applicable, for example, if the most important information from the results is the distance of the target form the camera sensor plane, depicted as the Z coordinate in Figure 4.2. It does not, however, work well in case there is a need to relate the resulting coordinates to other locations on or near the viewed plane.

Consequently, there is a second method for transforming the results to a more usable form. The localization system allows for easy setup of custom coordinate system within the viewed plane. The coordinate system is determined by four

points with coordinates $(0, 0), (0, 1), (1, 0)$ and $(1, 1)$, whose positions are marked in real world with four circular patterns. It is obvious therefore, that the location information here is only two dimensional. This is because when using a coordinate system setup by this method, it is assumed that all detected targets will lie in the viewed plane or negligibly close to it. As long as this assumption is correct, the third coordinate can be set to zero.

The four patterns are localized in the same way as described above and then they are used to define the new coordinate system. That means a transformation matrix is calculated, which projects a point from the camera centric coordinate system, to the one defined by the four points, as seen in Figure 4.3. This way, the target is localized within a user defined coordinate system, which every user can customize to the specific needs of his use case.

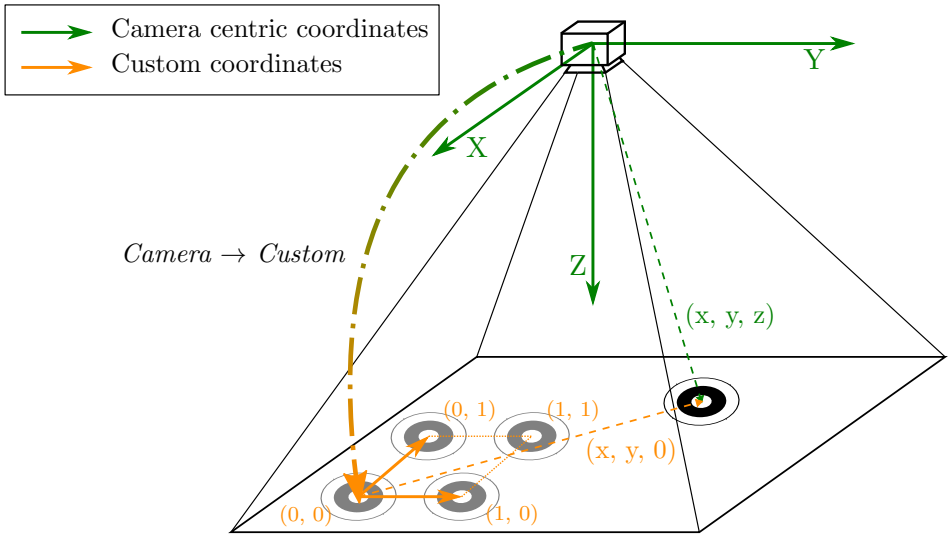


Figure 4.3: Transformation from camera centric coordinates to custom coordinate system

At this point, the localization system running on a single Android device is complete and ready to be used. Firstly, there is a simple user interface, which allows the user to calibrate the devices camera. Secondly, there are several parameters, the user is prompted to set, like the resolution of the camera and dimensions of the target circular pattern. Then the device can be positioned, so that the camera has the area of interest in its field of view. Optionally, a user can setup a custom coordinate system using several targets, which exactly fits his needs. Finally, the tracking can be started with a push of a button.

Chapter 5

Multiple Cameras

Field of view of a single camera can be very limited, especially when wide angle lenses are not usable on account of their high distortion. It is possible to extend the area covered by the camera by placing the camera further away, but this approach places big demands at resolution of the camera and therefore is practicable only up to a certain point. Not to mention that the expected indoors usage of the localization system may introduce limiting physical constraints. Analogically, keeping the camera at a fixed distance and downsizing everything in its field of view is possible only as far as the resolution is sufficient to perform the detection on the downscaled circular patterns. One of the possible solutions to this problem lies in using a whole set of cameras instead of just one.

This approach multiplies the field of view without the need of high resolution or with insatiable spatial requirements. On the other hand, it has higher computational requirements, because, instead of just one image at a time, there are multiple images—one from each camera—all of which have to be processed before moving on to the next frame. If all these cameras were to be attached to a single processing unit, it would have to be substantially powerful. This is not an issue once the tracking is already in progress and there is an *a priori* information about the position of the target, but in that case, all cameras, except the one which has the target in view, are redundant for the time being, so there is no need to even capture the images from these cameras.

The presented implementation does not work with a set of cameras but rather with a set of complete Android devices running the RobotTracker application and connected by a shared Wi-Fi network. This means there is an independent processing unit for each camera, which practically negates the need of a high performance central computer, as the actual processing of each image is done independently on

the other camera's images. However, there still needs to be a common mechanism assorting the results, tracking the target across all the cameras and controlling which camera (or cameras) should be capturing images at each given moment. Such a mechanism is especially desirable, because it cannot be assumed the devices have an infinite power source, but rather it has to be expected that they are powered by batteries with limited capacity. Selectively turning off the power consuming task of capturing and processing images helps prolong operational time of the localization system.

5.1 Spatial Arrangement

The usage of multiple cameras requires a predictable way of positioning the cameras above the area to be monitored. A square grid arrangement with the cameras at the junctions of the grid is a good option, since field of view of each of the cameras is a rectangle. That way, it can always be conclusively determined, which cameras are neighbours of each given camera and their respective position. Another advantage of such arrangement is that the information needed to describe the relative position of two cameras is reduced to their coordinates in this two dimensional grid.

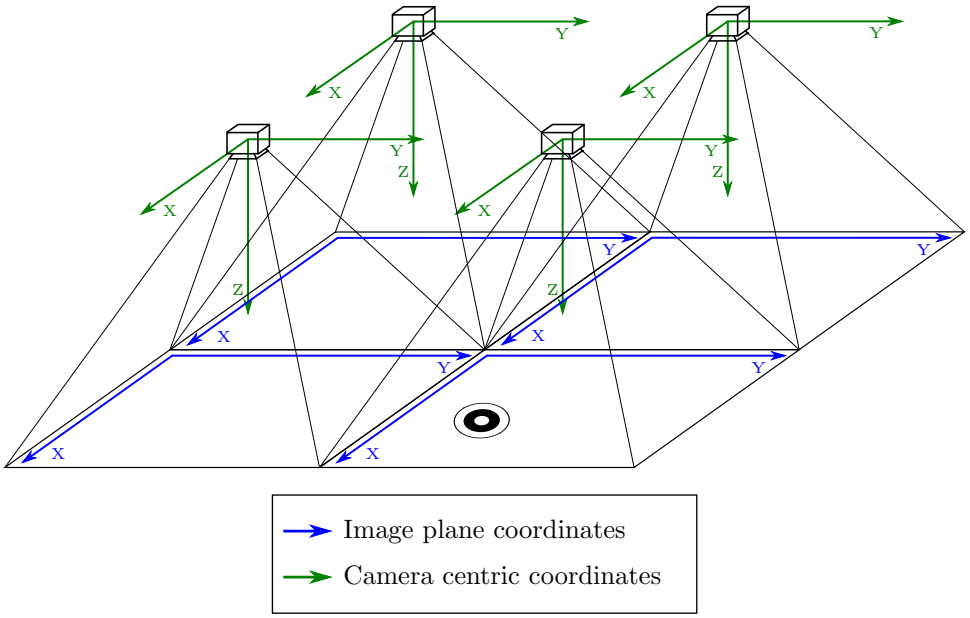


Figure 5.1: Multiple cameras spatial arrangement

To simplify the distance relationships between the cameras, one of them is chosen and it's position is used as an origin in the grid's coordinate system. Distances of all the other cameras are then defined as distances from this origin.

An obvious requirement, which needs to be met for a successful deployment, is that all the cameras have the same orientation. Otherwise, it would be impractically complicated to keep track of which edge of each frame is common to which neighbouring couple, or whether it is common to any neighbours at all. This is relatively easy to achieve on Android devices, however, because they all share the same default landscape camera orientation—ninety degrees counter-clockwise from the regular portrait orientation.

Therefore, when the cameras are being positioned, there are two pieces of information, which the system needs to get. Firstly, the relative position of the positioned camera in the grid of cameras, i.e. at which junction of the grid is this camera being positioned. Secondly, the distance (in the direction of both x and y axis) of the positioned camera from the camera at the origin in meters.

5.1.1 Field of View Overlap

To be able to recognize the target, the whole circular pattern needs to be within the image. If the cameras were to be positioned in such a way, that the field of view of the one camera started at exactly the same spot where the field of view of the next camera ended (as depicted in Figure 5.1), there would be a significant area on the borderline of the two fields of view, where the pattern would not be entirely in neither of them. Hence, during the time the target was in this area, it would not be possible to find it by any of those cameras and it would be lost. The problem is even worsened by the distortion and other optical imperfections of the lenses, which are usually worst near the edges of the field of view [17], although not that significantly after the undistortion procedure is carried out.

To counter this issue, it is judicious to let the fields of view of each pair of neighbouring cameras overlap somewhat. Just how much the fields of view should overlap depends on the size of the circular pattern and its ratio to the size of the area in the field of view of the camera. Practical experience show, that the width of the overlap area should be greater than the outer diameter of the circular pattern. The extra space is needed to allow some white space between the edge of the frame and the pattern itself, so that it can be recognized reliably by both neighbouring cameras. (To increase clarity of the illustrations, depiction of the fields of view overlap is omitted, except in Figure 5.2.)

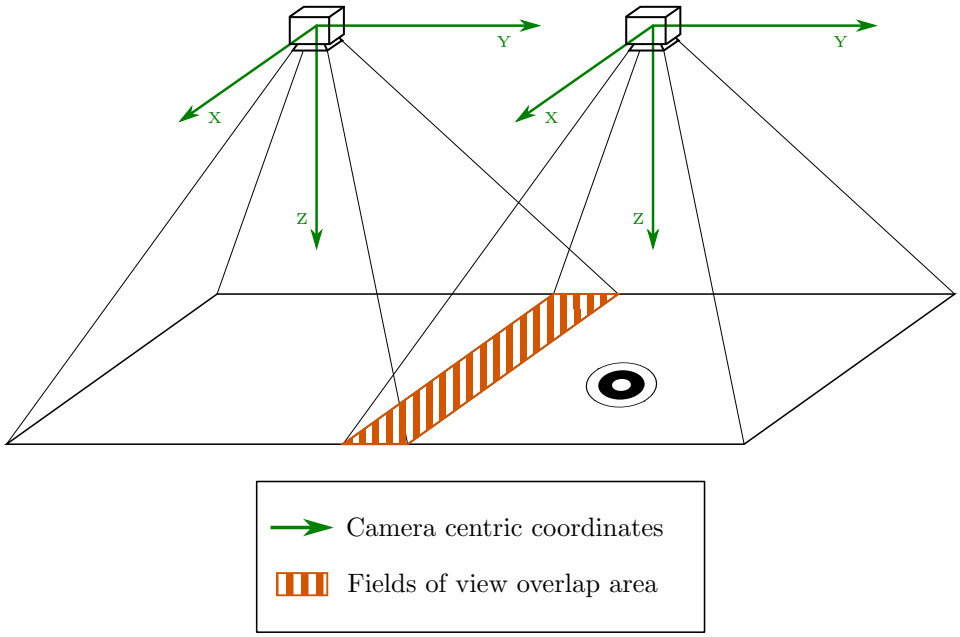


Figure 5.2: Overlap of the fields of view of neighbouring cameras

5.2 Coordinate Transformation

As stated before, each of the cameras is in this case actually a whole android device running the tracking application relatively independently on the others. When such an independent device tracks the target, it can output the results only in the original image plane coordinates or in its own camera centric coordinates, unique to each camera. There is no direct way however, to return the results in some globally understandable shape, so it is necessary to transform them.

The only information each of the separate devices has about the whole system is its own position in it, with respect to the device at the origin. The camera centric coordinate system of the camera at the origin can be therefore used as main coordinates to represent the results globally. They can also be used as a starting point for prospective further transformation (for example to the previously mentioned self defined custom coordinate system).

When using the square grid arrangement mentioned above, it is very simple to transform between local and main camera centric coordinates. The transformation is a simple translation in either positive or negative direction of the x and y axis of the camera centric coordinate system, depending on the camera's relative position

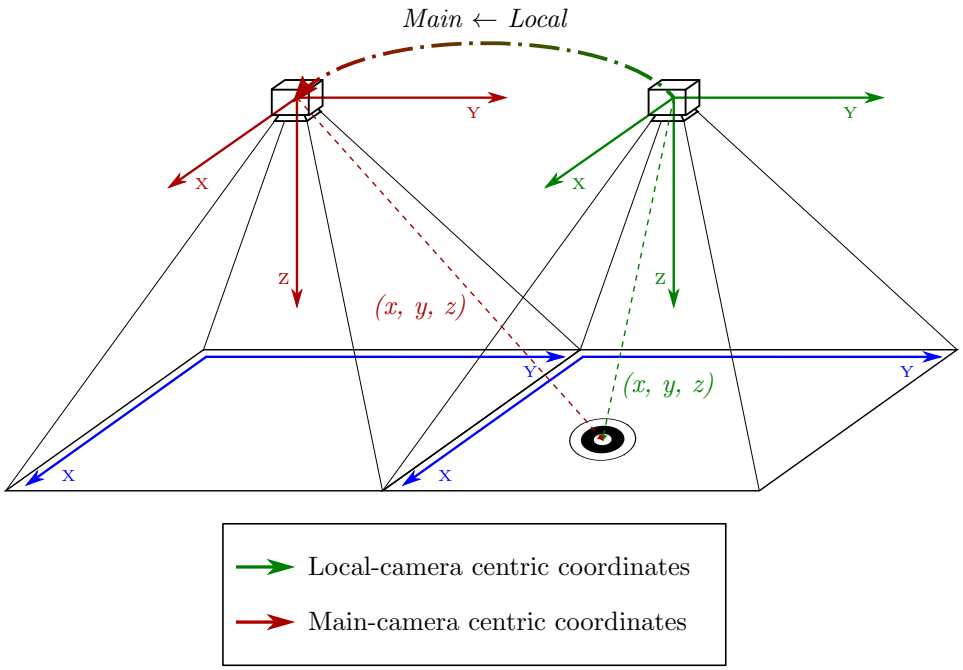


Figure 5.3: Transformation from local to main camera centric coordinates

in the grid (Figure 5.3). The transformed coordinates (belonging to the main camera) can then either be used directly, or for further transformation to a user defined custom coordinate system, as described in the previous chapter.

5.3 Selective Capturing

As mentioned before, it would not be practical to have all the devices capture frames all the time, because it is not likely that the target would be visible to every one of them. Therefore, it is convenient to pause the localization on devices, whose field of view the target leaves and only resume it, when the target re-enters it. However, when the localization and therefore the pattern recognition is not running, the device has no way of knowing, whether the target is re-entering its field of view, so it needs to obtain this information from an outside source.

To be able to do this, there is a device in the system, which is working as a server, which receives localization results from all other devices connected to it in a client-server relationship. In my implementation, one of the devices themselves serves as the server, receiving the results and controlling the clients. Generally

though, it can be any other device outside the group of capturing devices. It can even be a device, which is attached to the robot carrying the target, which is being localized, and which therefore can conveniently feed the results straight to its control unit.

Every time this server receives the result data from a client, it decides, whether that particular device should continue localizing, or if it should pause. Accordingly, it can notify some of the other clients, which are paused at the moment, that they should resume localizing, based on the same data from the client. These two decisions are made with two independent groups of condition checks. The checks are based on several parameters, which are precomputed by the clients when possible, to minimize the load on the server itself. The server only keeps a list of the positions of the clients available, which is created during setup, and a list of the clients which are currently localizing.

5.3.1 Resuming a Client

When the localization is started as a whole, it is expected that one of the devices has the target in its field of view. Only that device starts localizing and when it successfully localizes the target, it starts tracking it. Other devices are considered paused and they wait for the server to notify them to resume localizing. To make a decision to resume localization on a client, the target must be approaching an edge of the frame of the camera, which is currently tracking the target. The proximity of the target to the edge is determined by perpendicular distance of the target's centre from the edge in question. If the distance drops below a certain threshold distance, the target is considered to be near the edge.

Once it happens, the system compares several consecutive frames from this camera, evaluating, whether the distance of the target from the edge is monotonously receding. If it is the case, the system can make reasonable assumption that the target is about to leave the field of view of the current camera and also that it is about to enter or has already entered the field of view of the neighbouring camera, adjacent to that particular edge. Therefore, if such a camera actually is part of the system, i.e. if the neighbouring camera exists, it is notified to resume localization. In case the neighbouring camera does not exist, the system can only keep tracking the target with the current camera as long as possible.

For the system to be robust, the threshold distance has to be set correctly. Generally, the distance should be somewhat greater than the outer diameter of the circular pattern. If it were too small, the system might fail to notice the target is approaching the edge and subsequently fail to notify the adjacent camera to

Figure 5.4: Camera frame with a target near the edge

resume localization and lose track of the target altogether. On the other hand, if it were too big, there would be a lot of false alarms, resulting in resuming the localization in vain.

5.3.2 Pausing a Client

To make a decision to pause localization on a camera, two conditions have to be met. Firstly, it only makes sense to pause the localization, if the target is no longer inside the field of view of the camera. To eliminate possible temporary glitches, when the localization system fails to localize the target, even though it is still visible to the camera, there has to be no positive match in several consecutive frames. Only when the target is not found several times in a row, is it considered gone from the field of view and the first condition is met.

The second prerequisite for being able to pause a camera is the fact that at least one of the other cameras is active at the time. In standard situations, there always is another resumed camera, when the target leaves the field of view of the current one, because the only standard way for the target to leave the field of view is to come close to one of the edges. At least the camera adjacent to the edge the target came close to should therefore be active. However, in case this neighbouring camera does not exist, the target may have left the field of view and even though the system knew the target is about to depart, there was no device to resume localization on.

Other possible circumstances, in which the first condition would be satisfied, but without another device being active, may be caused by the view of target

being obstructed by an alien object or by an unexpectedly rapid change in position of the target. Either way, the best solution for the system is to keep the current camera localizing, for there are no inklings as to which camera should be preferred. Alternatively, the system could notify a random device to try and find the target once again, or it can even notify all of them. This approach, however, would be very wasteful as far as power is concerned and may put an unbearable load on the server, in case the system had been created with the use of a lot of devices.

5.4 Implementation

The extension of the original localization system Whycon [1] is implemented in the RobotTracker application for Android operating system. For increased clarity and to allow easier demonstration of the extension, the actual application works with a simplified version of the system. This version expects the set of devices to form a one dimensional array, creating a straight line arrangement, rather than a two dimensional grid arrangement. That way, setting the system up is easier, because the positioning of the individual devices does not have to be as precise and the revisory measurements are simpler to perform. Although this arrangement is not especially flexible, the covered area is increased greatly, compared to the original system.

The precision of the localization on the individual devices remains unchanged, the transformation from local to main camera centric coordinate system may, however, introduce some new inaccuracies. These inaccuracies originate from the distances between the cameras not being measured precisely or from the cameras not being aligned to be exactly parallel. Nevertheless, if the setup is carried out carefully, the system retains good precision, as the inaccuracies are not cumulative and are exclusive to each particular device.

On the other hand, under certain circumstances the multi device system has extra information, compared to the single point of view version. This happens when the target appear in fields of view of several cameras at once. In these situations, the system has either more localization results at the same time, or it receives the results with multiplied frequency, depending on the way the results are processed by the server. Another advantage of such cases is higher robustness of the target localization in environments with lots of obstacles. Obstacles which prevent the target from being viewed from one point of view are often unimportant for the view from other relevant components of the system.

Since all the results are actually just estimates of the real position of the tar-

get, it is therefore possible to get a better estimate, employing the probabilistic localization principles (as described in Chapter 2). However, when trying to take advantage of this feature of the localization system, it is a necessity to set it up accordingly. To get good results, there would have to be big overlaps of the camera's fields of view and slight modifications would have to be made to the way the system coordinates the operation of the cameras (e.g. the system should start with more than just one camera active, as there would likely be more than one camera with the target in view).

It is clear then, that emphasising the precision this way and therefore setting the system up with big overlaps between the field of view, goes directly against the goal of maximizing its spatial coverage. There is no simple way of pursuing both the spatial coverage and precision goals without one hampering the other. The implementation presented with this thesis thus chooses to focus on the goal of increased spatial coverage and puts lower priority on the precision aspect of multi-camera localization.

Chapter 6

Experiments

To illustrate the functionality of the presented system, I hereby present results of several experiments, which has been performed with it. The experiments were carried out using the devices listed in Table 6.1.

Device Name	CPU	GPU	RAM
Samsung Galaxy S4 Mini	2 × 1700 MHz, Krait	Adreno 305	1536 MB
HTC Desire	1000 MHz, Scorpion	Adreno 200	576 MB

Table 6.1: Testing devices

Most of the experiments were performed on the *Samsung*, as it better represents an average device of today. The more than 4 years old *HTC* was used only when cooperation of more devices is necessary.

6.1 Appropriate Setup Parameters

Before performing any experiments, it has to be made sure that the system has been setup the right way. Especially the multiple cameras variant of the system is quite sensitive in relation to good setup parameters. When setup incorrectly, the system may easily be rendered much less reliable. To avoid any issues, there are several parameters of the system, which a user of the system needs to consider.

First, there is the field of view overlap. For the system to be able to recognize the target at any given point in space, the whole target needs to be in view of one of the cameras. This means that when the target pattern touches an edge of the

camera frame and therefore starts leaving the field of view, it should already be wholly visible in the field of view of the neighbouring camera. If it is not the case, then, regardless of the fact, that every point on the viewed plane may be visible, there is an area near the edge of the frame, where the target cannot be localized.

This may not be a problem, as long as this area is small enough, but even then, the target may move inside this area along the edge and get completely lost to the system. Therefore, it is a good practice to place a pattern near the edge of the field of view to which we plan to add a neighbour and make sure the neighbour has the target in view as well when put in place.

The second parameter, which has to be kept in mind, is the threshold used to determine whether a target is near an edge. This is particularly important when the localized target may be expected to move rapidly. When they do, there is a greater danger of the target rushing out of view of an active camera before the system realizes what is happening. Such problems arise when the threshold is to small. On the other hand, when the threshold is too big, there might be a lot of unnecessary camera activations, which hinder the system performance.

It was empirically determined, that a good setting of this parameter is one and a half to two times the size of the target pattern, which in regular indoors means around 15% of the frame size. For applications with fast target movement, it is advisable to increase the threshold, for slow target applications, it may be considered decreasing it, to further increase resource efficiency of the system. (Please note, that the current implementation of the system, does not allow changes of this parameter at runtime, as it was found that the above mentioned value—15% of the frame size—works in majority of situations. There is a constant in the code, however, which can be changed as necessary.)

6.2 Performance

To evaluate general performance of the system, I measured the rate at which it is able to capture and process camera frames and produce results. Tests have been made both in the direct camera mode, in which the application captures the images itself and then goes straight on to process them, and in the file mode, in which the images are loaded from JPEG files, which had been manually captured beforehand. This illustrates what part of the time necessary to process one frame is spent loading it and what part is actually spent with localization.

All the experiments were done several times with diverse input data. Sometimes the target was closer to the camera, sometimes it was further away, the

backgrounds varied between uniform ones to quite complex and the lighting conditions were also altered throughout the testing. Average frame rates measured with the data and their standard deviations were then calculated separately for each resolution used.

Resolution:	800 × 600	640 × 480	400 × 300
Camera	11.22 ± 3.59	13.09 ± 4.41	–
Files	30.99 ± 8.40	43.92 ± 12.48	92.59 ± 26.77

Table 6.2: Average frames per second

Table 6.2 shows that the image retrieval part of the localization process takes up significant amount of time. This is caused both by the hardware limitations of the cameras, which in most cases allows a theoretical maximum of 30 FPS, and by the software limitations of Android operating system, which does not allow retrieval of the next frame at any specific time. Instead, it is necessary to dispatch a request for the next frame and then wait for a callback, which has quite unpredictable response times.

It is also clear from the standard deviations (listed alongside the frame rates in the Table 6.2) that the values fluctuate greatly in time. Therefore, it must be expected the frame rate may regularly reach values, which exceed the average by a third or even by a half grater. Due to this, the system may be rather unstable with a high resolution input, with which the peak processing times are scaled up accordingly.

Furthermore, in all the above experiments the system kept continuously tracking the target after its initial discovery. When facing a situation, in which the target is unexpectedly lost and found during tracking, even greater peak times result as the system needs to search the whole image, until it finds the target again.

To test this, an experiment was carried out on several series of frames, which contained an unexpectedly quick change of position of the target on every fourth frame. Figure 6.1 shows how the process times rapidly grow, when the tracking is interrupted. Standard deviations displayed in the graph illustrate, that the time needed to retrace the target is not only incomparably longer than the usual process time, but also quite unpredictable. Reducing the image resolution is thus a possible solution to this problem, although it only helps partially. A better way to prevent this problem is to maximize the general frame rate of the system, so that these situations are as scarce as possible.

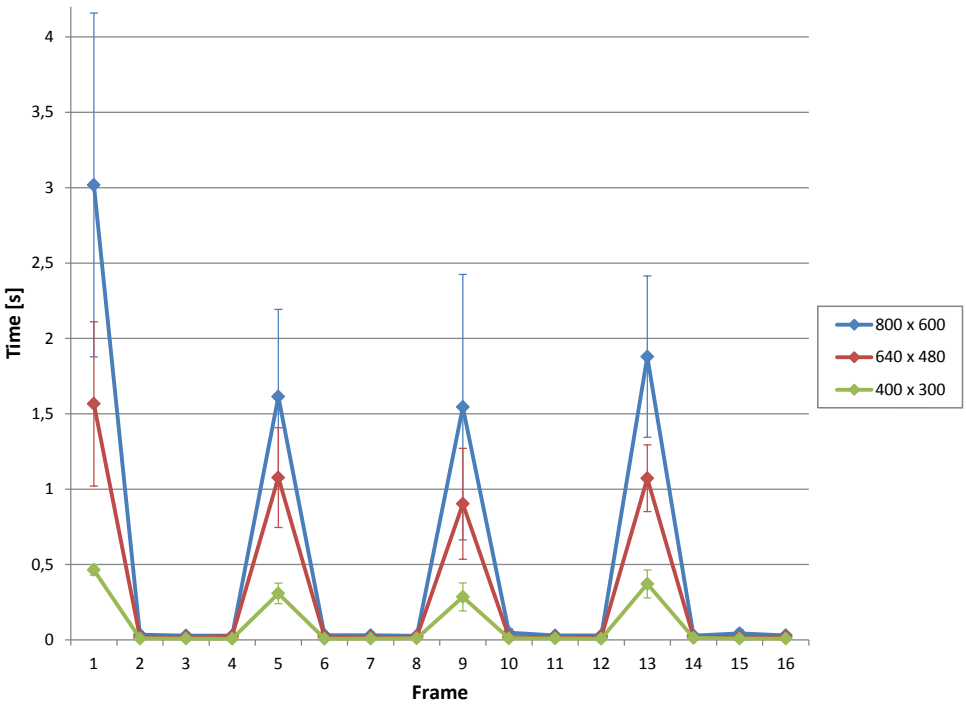


Figure 6.1: Frame processing times for interrupted data

However, one case in which the tracking interruption cannot be avoided is when the target is passing the border between fields of view of two neighbouring cameras. To illustrate, what happens in this scenario, an experiment was performed, in which the target starts in the field of view of the first device, then proceeds into view of the second one, comes back under the first device, and finally moves into view of the second one again.

Figure 6.2 shows the times needed to process each frame in the order, in which they arrived to the device, which was acting as a server. Peaks around frames 20, 55 and 110 represent the frames, in which the initial time-consuming target discovery was performed, when the target was entering (or re-entering) the field of view of the second camera, the first camera and the second camera again, respectively. The values measured between these peaks are mostly from one particular device, but, due to the overlap between the fields of view, just after each of the transitions there might be a few frames that still come from the previous camera. Note, that

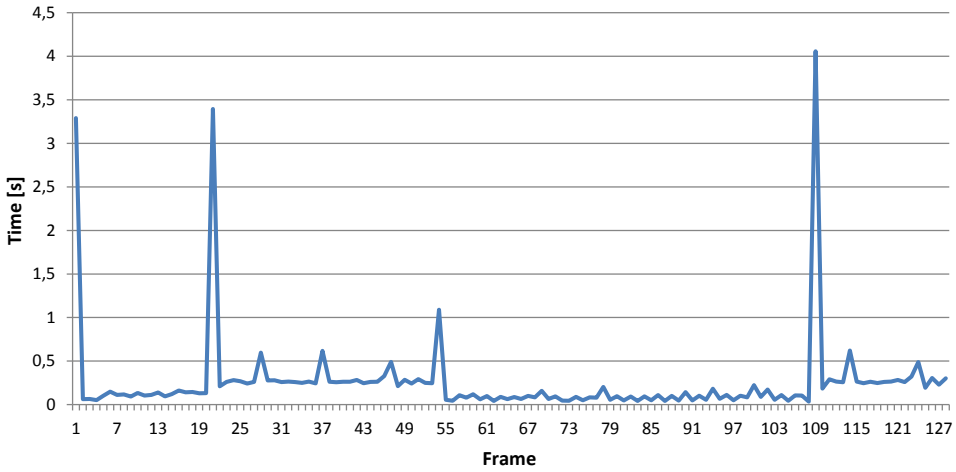


Figure 6.2: Frame processing times when transfering between cameras

the second device in this experiment was the older and much less powerful *HTC*, so its peak processing times just after transfer are even greater than they would be for an average present-day device.

To prevent this situation, which is not easily avoidable, from influencing the operation of the localization system, there must be a big enough overlap between the fields of view of the two cameras in question. That, together with the threshold triggering the localization at the second device early enough, allows the first device to continue tracking the target, while the second one performs the most time-intensive initial search.

6.3 System Precision

The most important feature of a localization system is its precision. To measure it, a custom coordinate system was created (as described in Chapter 4) with origin and axes aligned with the junctions of a tile floor. This provided a fairly precise reference frame, which could be used to evaluate the precision of the localization.

The actual experiment involved placing the target pattern precisely at the junctions of the tiles and localizing them. The localized target positions were then compared to the real position of the junction in terms of the same coordinate

system. The real position was determined with knowledge of the tiles joint to joint size—in this case 20.1 cm. The Camera 0 was placed some 2 meters above the floor near coordinates $(20, -20)$ and the Camera 1 was 66 cm in the positive direction of the x axis from the former.

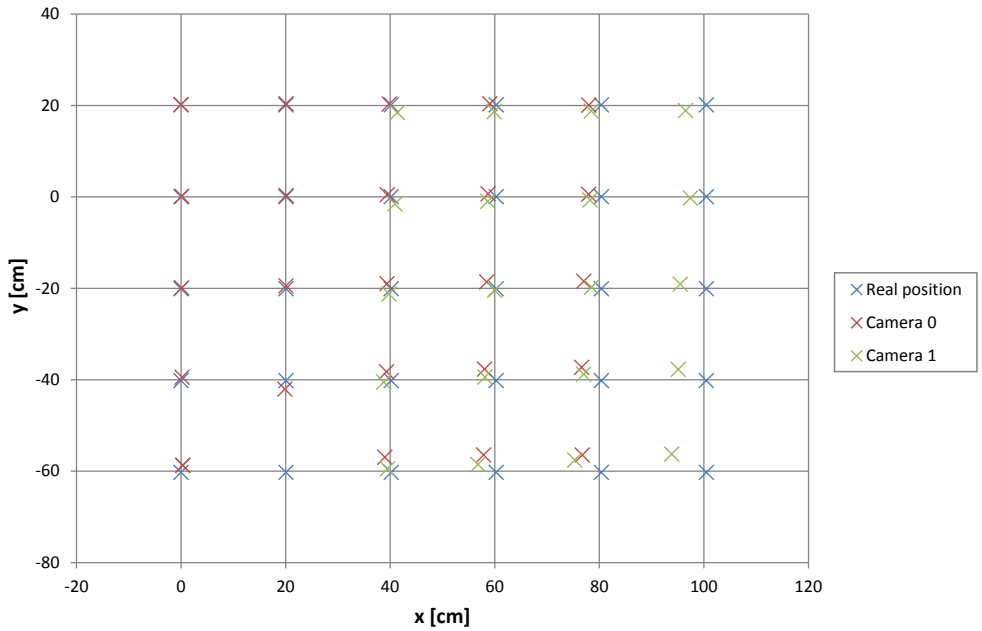


Figure 6.3: System precision

As can be seen in Figure 6.3, the precision of the system is quite good, although it gets worse the farther the target moves from the origin of the custom coordinate system. Hence, it may be advisable to place the four targets as far from each other as possible, while creating the custom coordinate systems, as this would reduce the cumulative error in the results transformation. The experiment also showed better accuracy near the centre of the frame as opposed to its edges, which can be credited to the imperfections of the pinhole camera model. Additionally, the figure reveals that the Camera 1 was calibrated very well and its localization is therefore subject to significant and increasing error, as the target gets near the edges of its field of view.

The second experiment testing the accuracy of the localization was performed with a target being dragged along a straight line parallel to one of the axes. The line went through the field of view of two neighbouring cameras, arranged the

same way as in the previous experiment. The results of the second experiment are shown in Figure 6.4 (please note, that the y axis scale has been enlarged for illustrative purposes, so the stray may seem exaggerated).

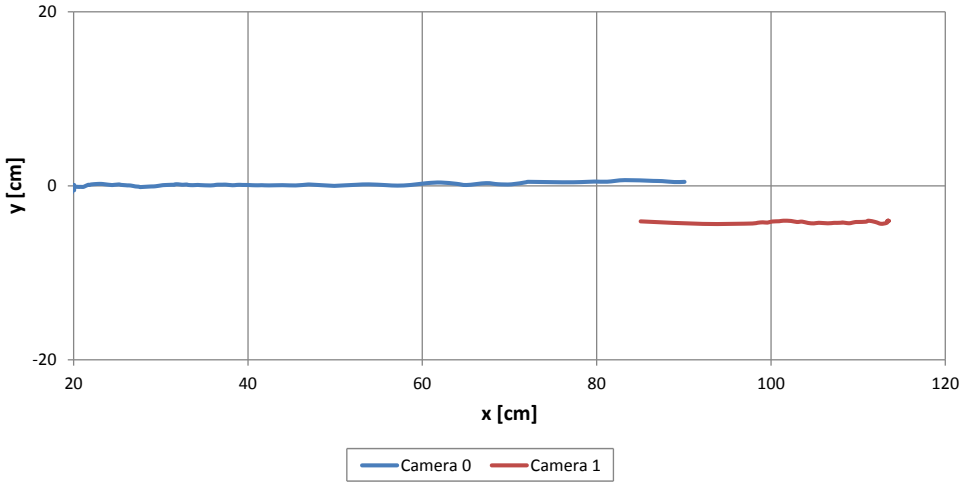


Figure 6.4: Target dragged along straight line

This experiment shows, that while the precision of the localization in itself remains constant, there is an offset in the results received from the second camera. This offset has been most likely caused by imprecise alignment of the two cameras with the created custom coordinate system. Note also, that even though the target moved in the positive direction along the x axis, the graph says nothing about the order, in which the results were received. The first results from the second camera were actually measured earlier than the last results from the first, so the difference in the x coordinate of the end of the blue line and the start of the red line is not an inaccuracy. All in all, it can be said, that with moderate care when placing the cameras, especially making sure they are aligned and parallel, it is possible to have a consistent precision across the whole system of multiple devices.

Chapter 7

Conclusion

The presented thesis deals with a camera-based localization system for mobile robots. Instead of regular cameras, this instance of the system works with complete handheld devices running the Android operating system. This switch allowed for creating a multiple point of view system of mutually interconnected devices (cameras), extending the capabilities of the system.

The proposed system implementation consists of a single Android application, with a simple user interface, which guides the user through the procedures needed to calibrate the camera and to setup the system itself. The application implementation further entails an adaptation of the Whycon localization system, extended of the capability to take advantage of a common Wi-Fi network to interconnect more devices, giving the system additional points of view.

Although it is certainly possible to take advantage of the multiple viewpoints in many different ways, such as increasing the reliability or precision of the whole system, it has been the main goal of this thesis to increase the spatial coverage. Therefore, it is suggested, that the devices are placed above the monitored area, in which robots should be localized, forming a uniform square grid or a line.

Several routines are incorporated in the system, which help the system track a target across the fields of view of all the devices, switching of the cameras not needed at any given point, until such time, when they are needed again. Android devices being mostly battery powered, the above is especially useful, for it helped making the system more power efficient aside from giving it better responsiveness and reducing the number of dispensable failed localization attempts.

The experiments verified, that the system is indeed usable and that it fulfils the goal of allowing localization within an expanded area. Although speed of the

system is not overwhelming, it is sufficient even on average hardware available today. Nevertheless, the system still has aspects, which deserve future review. For example the current OpenCV library includes an alternative, much faster and much more convenient way of accessing the camera frames. However, it does not reliably work across the wide spectrum of available devices.

Another improvement may lie in extending the capabilities of the system further, to be able to track more targets at once. This would involve solving issues regarding distinguishing between the individual targets and more complex mechanism controlling the operation of the redundant parts of the system. Changes to the base target recognition would probably be necessary to accomplish these goals. A different set of opportunities for enhancements lies in simplification of the setup process, for example when creating the coordinate system, or while positioning the individual devices into the grid. A method, which would automatically determine the relative position and alignment of the cameras during setup, would be very convenient, as it would eliminate a lot of inaccuracies, caused by external measurements of the devices' exact positions.

Overall though, the system is capable of quite precise (within a centimetre) localization of a mobile robot by means of finding the circular pattern and it does so in an area, limited in size only by the number of devices used and the range of the used Wi-Fi network.

Bibliography

[1] Tomáš Krajník, Matias Nitsche, Jan Faigl, Marta Mejail, Libor Přeučil, and Tom Duckett. External localization system for mobile robotics. In *16th International Conference on Advanced Robotics (ICAR 2013)*, pages 25–29. IEEE, 2013.

[2] Jan Faigl, Tomáš Krajník, Jan Chudoba, Libor Přeučil, and Martin Saska. Low-cost embedded system for relative localization in robotic swarms. In *ICRA*, pages 993–998. IEEE, 2013.

[3] Marek Skalka. Srovnání lokalizačních technik. Master's thesis, Univerzita Karlova v Praze, 2011. (In Czech).

[4] Rudy Negenborn. Robot localization and kalman filters: On finding your position in a noisy world. Master's thesis, Utrecht University, 2003.

[5] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents series. MIT Press, 2005.

[6] J. Borenstein, H.R. Everett, and L. Feng. *Where Am I? Sensors and Methods for Mobile Robot Positioning*. The University of Michigan, 1996.

[7] M.M. Morrison. Inertial measurement unit, December 1987. US Patent 4711125.

[8] M. W M G Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, June 2001.

[9] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, October 2005.

[10] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[11] Reinhard Klette. *Concise Computer Vision: An Introduction into Theory and Algorithms*. Springer, London, 2014.

[12] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly Media Inc., 2008.

[13] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106–1112, June 1997.

[14] OpenCV. OpenCV Documentation. `http://docs.opencv.org/`, 2014. (Online; accessed on 6 May 2014).

[15] E.W. Jin. Image quality quantification in camera phone applications. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 5336–5339, March 2008.

[16] Shaowu Yang, Sebastian A. Scherer, and Andreas Zell. An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, 69(1–4):499–515, 2013.

[17] Alexandros Gavriilidis, Carsten Stahlschmidt, Joerg Velten, and Anton Kummert. Evaluation of the lens distortion errors in 2-d image to 3-d world coordinate projections. In *Multidimensional Systems (nDS), 2013. Proceedings of the 8th International Workshop on*, pages 1–5, September 2013.

Appendix A

CD content

A CD is attached to the printed version of this work containing the text of the thesis in a PDF format, the source code of the thesis in LaTeX format and the source code of my implementation of the localization system—the android application RobotTracker. The directory structure on the CD is described in the following table.

Directory \ File	Description
`thesis.pdf`	the bachelor thesis report in PDF format
`thesis_src`	the bachelor thesis source code in LaTeX format
`RobotTracker`	the Eclipse project directory of the RobotTracker Android application, complete with the source code and other resources

Table A.1: Directory structure on the CD